

**OBJECT DETECTION USING TENSOR FLOW**

*Submitted by*

**Yogesh Kumar Maurya**

**Registration No.: 11715990**

**Course Name: INT248**

**Section: KOE64**

*Submitted to*

**Usha Mittal Ma'am**

**(Assistant Professor)**

**School of Computer Science & Engineering Lovely Professional University, Phagwara (Aug- Dec, 2020)**

**INTRODUCTION:-**

An object detection model is trained to detect the presence and location of multiple classes of objects. For example, a model might be trained with images that contain various pieces of fruit, along with a *label* that specifies the class of fruit they represent (e.g. an apple, a banana, or a strawberry), and data specifying where each object appears in the image.

When an image is subsequently provided to the model, it will output a list of the objects it detects, the location of a bounding box that contains each object, and a score that indicates the confidence that detection was correct.

In this,we will describe the signature for Single-Shot Detector models converted to TensorFlow Lite from the TensorFlow Object Detection API.

## ****What is Object Detection?****

Object Detection is the process of finding real-world object instances like car, bike, TV, flowers, and humans in still images or Videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

Object Detection can be done via multiple ways:

* Feature-Based Object Detection
* Viola Jones Object Detection
* SVM Classifications with HOG Features
* Deep Learning Object Detection

### Input Signature

The model takes an image as input.

Lets assume the expected image is 300x300 pixels, with three channels (red, blue, and green) per pixel. This should be fed to the model as a flattened buffer of 270,000 byte values (300x300x3). If the model is [quantized](https://www.tensorflow.org/lite/performance/post_training_quantization), each value should be a single byte representing a value between 0 and 255.

You can take a look at our [example app code](https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android) to understand how to do this pre-processing on Android.

### Output Signature

The model outputs four arrays, mapped to the indices 0-4. Arrays 0, 1, and 2 describe N detected objects, with one element in each array corresponding to each object.

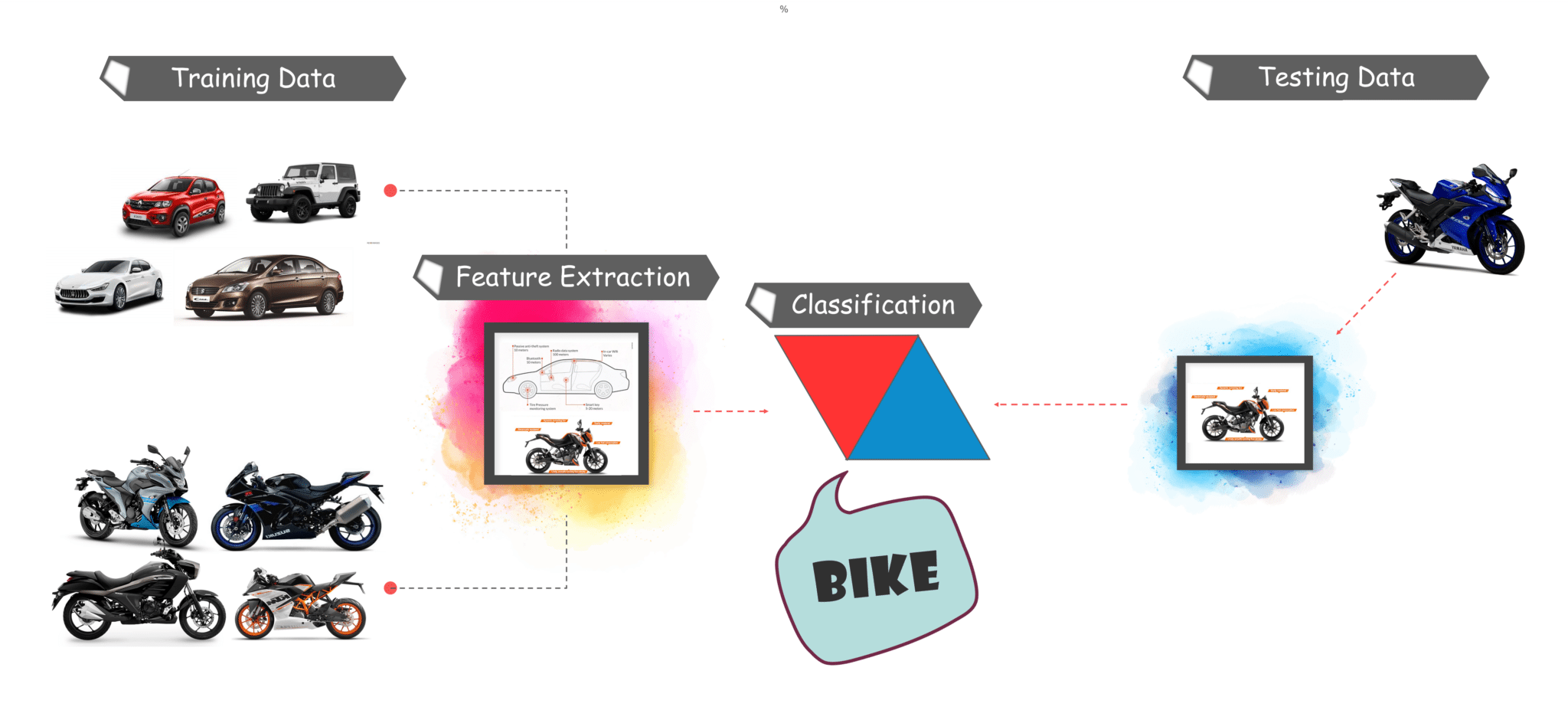
| **Index** | **Name** | **Description** |
| --- | --- | --- |
| 0 | Locations | Multidimensional array of [N][4] floating point values between 0 and 1, the inner arrays representing bounding boxes in the form [top, left, bottom, right] |
| 1 | Classes | Array of N integers (output as floating point values) each indicating the index of a class label from the labels file |
| 2 | Scores | Array of N floating point values between 0 and 1 representing probability that a class was detected |
| 3 | Number of detections | Integer value of N |

## ****OUTPUTS:-****

## Object detection | TensorFlow Lite Live Object Detection. Live Object Detection using the… | by Gilbert Tanner | Towards Data Science ****Object Detection Workflow****

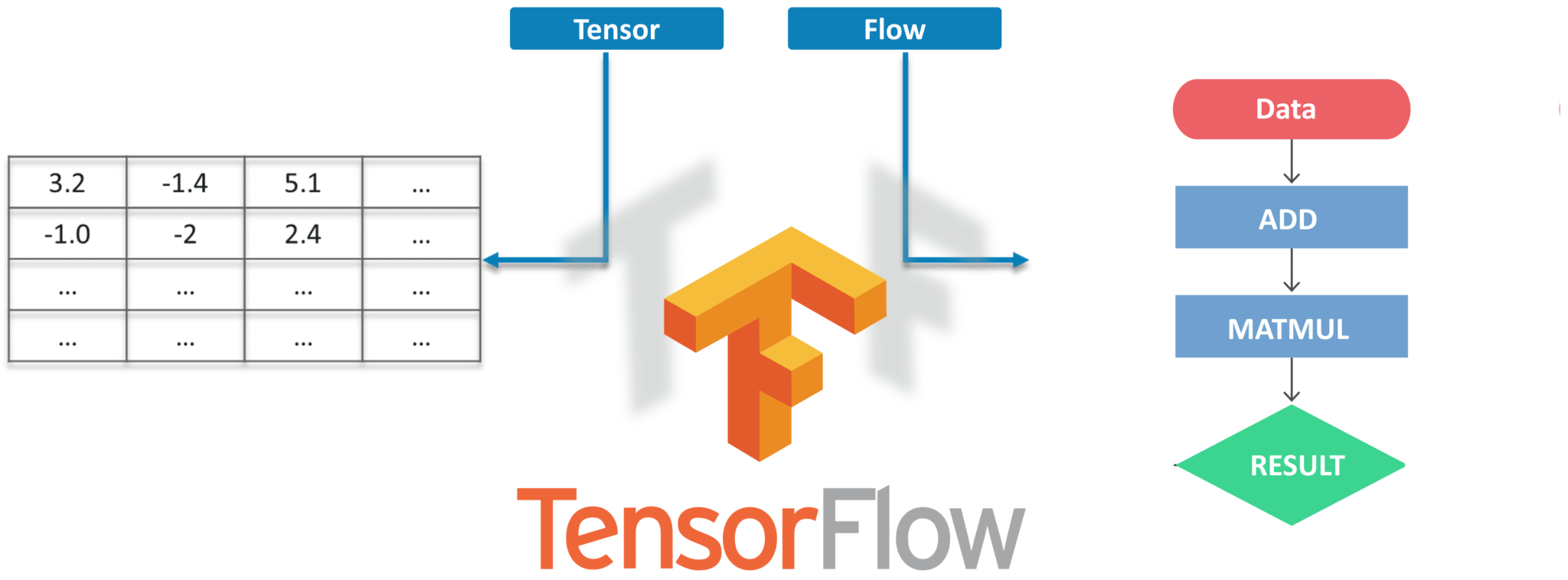
Every Object Detection Algorithm has a different way of working, but they all work on the same principle.

**Feature Extraction:**They extract features from the input images at hands and use these features to determine the class of the image. Be it through MatLab, Open CV, Viola Jones or Deep Learning.



## ****What is TensorFlow?****

Tensorflow is Google’s Open Source Machine Learning Framework for dataflow programming across a range of tasks. Nodes in the graph represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (**tensors**) communicated between them.



Tensors are just multidimensional arrays, an extension of 2-dimensional tables to data with a higher dimension. There are many features of Tensorflow which makes it appropriate for Deep Learning. So, without wasting any time, let’s see how we can implement Object Detection using Tensorflow.

**STEPS FOR OBJECT DETECTION**

### ****Getting Prerequisites****

* Before working on the Demo, let’s have a look at the prerequisites. We will be needing:
  + Python
  + TensorFlow
  + Tensorboard
  + [Protobuf v3.4 or above](https://github.com/google/protobuf/releases)

### ****Setting up the Environment****

* Now to Download TensorFlow and TensorFlow GPU you can use pip or conda commands:
* For all the other libraries we can use pip or conda to install them. The code is provided below:
* Next, we have Protobuf: **Protocol Buffers**(Protobuf)  are Google’s language-neutral, platform-neutral, extensible mechanism for serializing structured data, – think of it like XML, but smaller, faster, and simpler. You need to [***Download Protobuf***](https://github.com/google/protobuf/releases) version 3.4 or above for this demo and extract it.
* Now you need to Clone or Download TensorFlow’s Model from ***[Github](https://github.com/tensorflow/models" \t "_blank)***. Once downloaded and extracted rename the “models-masters” to just “**models**“.
* Now for simplicity, we are going to keep “models” and “protobuf” under one folder “**Tensorflow**“.
* Next, we need to go inside the Tensorflow folder and then inside research folder and run protobuf from there using this command:
* To check whether this worked or not, you can go to the **protos** folder inside models>object\_detection>protos and there you can see that for every proto file there’s one python file created.
* After the environment is set up, you need to go to the “**object\_detection**” directory and then create a new python file. You can use **Spyder** or **Jupyter** to write your code.
* Next, we will download the model which is trained on the [***COCO dataset***](http://cocodataset.org/#home). COCO stands for **Common Objects in Context,**this dataset contains around 330K labeled images. Now the model selection is important as you need to make an important tradeoff between **Speed and Accuracy**. Depending upon your requirement and the system memory, the correct model must be selected.

## ****Live Object Detection Using Tensorflow****

For this Demo, we will use the same code, but we’ll do a few **tweakings**. Here we are going to use ***OpenCV*** and the camera Module to use the live feed of the webcam to detect objects.

* Add the OpenCV library and the camera being used to capture images. Just add the following lines to the import library section.
* Next, we don’t need to load the images from the directory and convert it to numPy array as OpenCV will take care of that for us

### ****APPLICATIONS:-****

### ****1.Facial Recognition****

### 2.****People Counting****

### ****3.Industrial Quality Check****

### ****4.Self Driving Cars****

## Conclusion

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

.